```
import tkinter as tk
from tkinter import messagebox
import openai
import os

openai.api_key = os.getenv("OPENAI_API_KEY")

SYSTEM_PROMPT = """
You are a strategic advisor based on the teachings of Sun Tzu's The Art of War.
Your task is to interpret any situation (social, workplace, political, psychological)
and respond in modern, clear writing.

Always use this structure in your replies:
 Situation Analysis:
[Explain the true nature of the conflict.]

⚖ Strengths and Weaknesses:
[Analyze both sides.]

 Strategic Objective:
[State the goal the user should aim for.]

 Recommended Strategy:
[2–3 practical actions, grounded in Sun Tzu’s philosophy.]

 Quote for Reflection:
[A single, relevant quote from The Art of War.]

Write in a modern, empathetic tone with clear logic, but make sure the strategy
comes from Sun Tzu’s principles.
"""


def analyze_situation(user_input):
    return (
        " Situation Analysis:\nYou are facing a challenge that requires strategic
awareness.\n\n"
        "⚖ Strengths and Weaknesses:\nAssess your own strengths and be mindful
of your opponent's vulnerabilities.\n\n"
        " Strategic Objective:\nSecure your position through preparation, subtlety,
and timing.\n\n"
        " Recommended Strategy:\n1. Remain calm and observe the opponent.\n2.
Prepare your response in private.\n3. Strike only when your opponent
overextends.\n\n"
        " Quote for Reflection:\n'Victorious warriors win first and then go to war,
while defeated warriors go to war first and then seek to win.'"
    )

def gpt_strategy_advice(user_input, conflict_type):
```

```
    try:
        response = openai.ChatCompletion.create(
            model="gpt-4",
            messages=[
                {"role": "system", "content": SYSTEM_PROMPT},
                {"role": "user", "content": f"Conflict Type: {conflict_type}\n\
nSituation:\n{user_input}"}
            ],
            temperature=0.7,
            max_tokens=500
        )
        return response["choices"][0]["message"]["content"].strip()
    except Exception as e:
        return f"(GPT offline — using fallback engine)\n\
n{analyze_situation(user_input)}"

def on_click():
    user_text = input_box.get("1.0", tk.END).strip()
    if not user_text:
        messagebox.showwarning("Input Needed", "Please describe your
situation.")
        return
    conflict = conflict_type.get()
    use_gpt_flag = use_gpt.get()
    if use_gpt_flag:
        strategy = gpt_strategy_advice(user_text, conflict)
    else:
        strategy = analyze_situation(user_text)
    output_box.delete("1.0", tk.END)
    output_box.insert(tk.END, strategy)
    # Save to file
    with open("strategy_output.txt", "w", encoding="utf-8") as f:
        f.write("Situation:\n")
        f.write(user_text + "\n\n")
        f.write("Strategy:\n")
        f.write(strategy)

# GUI setup
root = tk.Tk()
root.title("Sun Tzu Strategy Advisor")
root.geometry("600x700")

label = tk.Label(root, text="Describe your situation:", font=("Helvetica", 12))
label.pack()

input_box = tk.Text(root, height=10, width=70)
input_box.pack()
```

```
tk.Label(root, text="Type of Conflict:", font=("Helvetica", 10)).pack()
conflict_type = tk.StringVar(value="Workplace")
conflict_menu = tk.OptionMenu(root, conflict_type, "Social", "Workplace",
"Legal", "Family", "Business", "Other")
conflict_menu.pack()

use_gpt = tk.BooleanVar()
gpt_checkbox = tk.Checkbutton(root, text="Use GPT Engine", variable=use_gpt)
gpt_checkbox.pack()

submit_button = tk.Button(root, text="Generate Strategy", command=on_click)
submit_button.pack()

output_box = tk.Text(root, height=25, width=70)
output_box.pack()

root.mainloop()
```